

A Unifying Framework for Typical Multi-Task Multiple Kernel Learning Problems

Cong Li, Michael Georgiopoulos and Georgios C. Anagnostopoulos

congli@eecs.ucf.edu, michaelg@ucf.edu and georgio@fit.edu

Abstract

Over the past few years, Multi-Kernel Learning (MKL) has received significant attention among data-driven feature selection techniques in the context of kernel-based learning. MKL formulations have been devised and solved for a broad spectrum of machine learning problems, including Multi-Task Learning (MTL). Solving different MKL formulations usually involves designing algorithms that are tailored to the problem at hand, which is, typically, a non-trivial accomplishment.

In this paper we present a general Multi-Task Multi-Kernel Learning (Multi-Task MKL) framework that subsumes well-known Multi-Task MKL formulations, as well as several important MKL approaches on single-task problems. We then derive a simple algorithm that can solve the unifying framework. To demonstrate the flexibility of the proposed framework, we formulate a new learning problem, namely Partially-Shared Common Space (PSCS) Multi-Task MKL, and demonstrate its merits through experimentation.

Keywords: Multiple Kernel Learning, Multi-task Learning, Support Vector Machines

1 Introduction

Kernel methods play an important role in machine learning due to their elegant property of implicitly mapping samples from the original space into a potentially infinite-dimensional feature space, in which inner products can be calculated directly via a kernel function. It is desired that samples are appropriately distributed in the feature space, such that kernel-based models, like Support Vector Machines (SVMs) [4] [20], could perform better in the feature space than in the original space. Since the feature space is implicitly defined via the kernel function, it is important to choose the kernel appropriately for a given task. In recent years, substantial effort has been devoted on how to learn such kernel function (or equivalently, kernel matrix) from the available data. According to the Multi-Kernel Learning (MKL) approach [13], which is one of the most popular strategies for learning kernels, multiple predetermined kernels are, most commonly, linearly combined. Subsequently, in lieu of tuning kernel parameters via some validation scheme, the combination coefficients are adapted to yield the optimal kernel in a data-driven manner. A thorough review of MKL methods and associated algorithms is provided in [7].

So far, several MKL formulations, along with their optimization algorithms, have been proposed. The earlier work in [13] suggests a MKL formulation with trace constraints over the linearly combined kernels, which is further transformed into a solvable Semi-Definite Programming problem. Also, Simple-MKL [14] relies on a MKL formulation with L_1 -norm constrained coefficients. In each iteration of the algorithm, a SVM problem is solved by taking advantage of an existing efficient SVM solver and the coefficients are updated via a gradient-based scheme. A similar algorithm is also applied in [23]. In [11], an L_2 -norm constraint is applied to the linear combination coefficients and the proposed min-max formulation is transformed into a Semi-Infinite Programming (SIP) problem, which is then solved via a cutting plane algorithm. Moreover, two algorithms are proposed in [12] to solve the L_p -MKL formulation, where the coefficient constraint is generalized to an L_p -norm constraint. The relationship between the latter L_p -MKL formulation and one

that entails a Group-Lasso regularizer is pointed out in [24], which utilizes a block coordinate descent algorithm to solve the problem in the primal domain. Besides MKL-based SVM models, a MKL formulation for Kernel Ridge Regression (KRR)[16] has been proposed in [3]. Cast as a min-max problem, it is solved via an interpolated iterative algorithm, which takes advantage of the closed-form solution for the KRR kernel coefficients.

Furthermore, MKL has been applied to Multi-Task Learning (MTL) giving rise to Multi-Task Multi-Kernel Learning (Multi-Task MKL) approaches, where several tasks with shared feature representations are simultaneously learned. As was the case with single-task MKL approaches mentioned in the previous paragraph, most Multi-Task MKL formulations that have appeared in the literature are accompanied by an algorithm tailored to their particular formulation. For example, a framework is formulated in [18], where individual tasks utilize their own task-specific space in conjunction with a shared space component, while the balance between these two components is controlled through weights. Subsequently, an exchange algorithm is proposed to solve the resulting min-max problem. Additionally, in [1], a Group-Lasso regularizer on SVM weights is considered that yields coefficient sparsity within a group of tasks and non-sparse regularization across groups. The resulting problem is solved in the dual domain via a mirror descent-based algorithm. Also, in [15], the Group-Lasso regularizer is generalized to $L_p - L_q$ regularization and two optimization problems are addressed, one being convex and the other non-convex. Next, for each of these two formulations, a specialized algorithm is proposed. Furthermore, in [22], Multi-Task MKL is formulated in the primal domain by penalizing the SVM weight of each task to be close to a shared common weight. Finally, maximum entropy discrimination is employed in [9] to construct a Multi-Task MKL framework.

Proposing useful MKL formulations and devising effective algorithms specifically tailored to solving them is, in most cases, a non-trivial task; such an endeavor requires fair amounts of insight and ingenuity. This is even more so true, when one considers applying MKL to MTL problems. In this paper, we present a general kernel-based Multi-Task MKL framework that subsumes well-known Multi-Task MKL formulations, as well as several important MKL approaches on single-task machine learning problems. Its unifying character stems from its applicability to prominent kernel-based tasks, such as SVM-based binary classification, KRR-based regression and outlier detection based on One-Class SVM [17] and Support Vector Domain Description (SVDD) [19], to name a few important ones. Also, it accommodates various feature space sharing approaches that may be encountered in a Multi-Task MKL setting, as well as single-task MKL settings with different constraints. For example, it subsumes L_2 -MKL and L_p -MKL considered in [11] and [12], the KRR-based MKL in [3] and the Multi-Task MKL formulations in [18] and [15]. We introduce our framework in Section 2.

In Section 3 we state an equivalency between solving SIP problems and Exact Penalty Function (EPF) optimization, which may be viewed as a useful result on its own. As the new framework can be cast as a SIP problem, the previous equivalency allows us in Section 4 to eventually derive a straightforward and easy to implement algorithm for solving the new EPF-based formulation. For a given Multi-Task MKL problem that is a special case of our framework, the availability of our algorithm eliminates the need of using an existing, potentially complicated, algorithm or deriving a new algorithm that is tailored to the problem at hand. A further major advantage of the algorithm is that it can leverage from already existing efficient kernel machine solvers (*e.g.*, SVM solvers) or closed-form solutions (*e.g.*, KRR solution) for given problems.

In Section 5, we present a new specialization of our framework, called Partially-Shared Common Space (PSCS) Multi-Task MKL. It permits some tasks to share a common feature space, while other tasks are allowed to utilize their own feature space. This approach follows the spirit of recent MTL research, which allows tasks to be grouped, while tasks belonging to the same group can share information with each other. Some examples include the works in [25], [8], [10] and [26], to name a few. This is a generalization of the traditional MTL setting, according to which all tasks are considered as one group and are learned concurrently. Our PSCS Multi-Task MKL formulation is a combination of the grouped MTL setting and MKL, which appreciably differs from previously cited works. The new formulation concretely showcases the generality of the proposed Multi-Task MKL framework, the flexibility in choosing a suitable Multi-Task MKL feature sharing strategy, as well as the usefulness of the derived algorithm. The merits of PSCS are illustrated in Section 6, where we compare its performance on classification benchmark data sets against the common-space (CS) and independent-space (IS) alternatives. Proofs of our analytical work are given in the appendices.

2 Problem Formulation

Consider a supervised learning task with parameter α , which can be expressed in the form

$$\max_{\alpha \in \Omega(\alpha)} \bar{g}(\alpha, \mathbf{K}) \quad (1)$$

where \mathbf{K} is the kernel matrix of the training data set, *i.e.*, its (i, j) entry is $K(\mathbf{x}_i, \mathbf{x}_j) \triangleq \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$, and \mathcal{H} is the Hilbert space reproduced by the kernel function k . Here, ϕ is the feature mapping that is implied by the kernel function $k(\cdot, \cdot)$ and $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, where \mathcal{X} is an input set. Also, suppose \bar{g} has a finite maximum, a finite number of local maxima with respect to α in the feasible set $\Omega(\alpha) \subset \mathbb{R}^n$ and is affine with respect to the individual entries of \mathbf{K} . Some well-known supervised learning tasks that feature these characteristics are considered in SVM, KRR, SVDD and One-Class SVM. Their dual-domain objective functions \bar{g} for n training samples are given in Equation (2) through Equation (5) respectively.

$$\bar{g}_{SVM}(\alpha, \mathbf{K}) \triangleq \alpha' \mathbf{1} - \frac{1}{2} \alpha' \mathbf{Y} \mathbf{K} \mathbf{Y} \alpha, \quad (2)$$

$$\text{with } \Omega(\alpha) \triangleq \{\alpha \in \mathbb{R}^n : \mathbf{0} \preceq \alpha \preceq C \mathbf{1}, \alpha' \mathbf{y} = 0\}$$

$$\bar{g}_{KRR}(\alpha, \mathbf{K}) \triangleq 2\alpha' \mathbf{y} - \alpha' (\lambda \mathbf{I} + \mathbf{K}) \alpha, \quad (3)$$

$$\text{with } \Omega(\alpha) \triangleq \mathbb{R}^n$$

$$\bar{g}_{SVDD}(\alpha, \mathbf{K}) \triangleq \alpha' \mathbf{k} - \alpha' \mathbf{K} \alpha, \quad (4)$$

$$\text{with } \Omega(\alpha) \triangleq \{\alpha \in \mathbb{R}^n : \mathbf{0} \preceq \alpha \preceq C \mathbf{1}, \alpha' \mathbf{1} = 1\}$$

$$\bar{g}_{OSVM}(\alpha, \mathbf{K}) \triangleq -\alpha' \mathbf{K} \alpha, \quad (5)$$

$$\text{with } \Omega(\alpha) \triangleq \{\alpha \in \mathbb{R}^n : \mathbf{0} \preceq \alpha \preceq \frac{1}{\nu l} \mathbf{1}, \alpha' \mathbf{1} = 1\}$$

In the above examples, $'$ signifies transposition vector/matrix, $\mathbf{y} \triangleq [y_1, \dots, y_n]'$ is the target vector, $\mathbf{k} \triangleq [k(\mathbf{x}_1, \mathbf{x}_1), \dots, k(\mathbf{x}_n, \mathbf{x}_n)]'$ and $\mathbf{Y} \triangleq \text{diag}(\mathbf{y})$, where $\text{diag}(\cdot)$ is an operator yielding a diagonal matrix, whose diagonal is formed by the operand's vector argument. Additionally, C, λ, ν, l are scalar non-negative regularization parameters for the corresponding learning problems and $\mathbf{1}$ is the all-ones vector of appropriate dimension.

Assume, now, that we have T such tasks and let $\{\mathbf{x}_i^t, y_i^t\}$, $i = 1, \dots, N_t$, be the training data for task t , where $t = 1, \dots, T$. Let $k_m(\mathbf{x}_i^t, \mathbf{x}_j^t) \triangleq \langle \phi_m(\mathbf{x}_i^t), \phi_m(\mathbf{x}_j^t) \rangle_{\mathcal{H}_m}$, where $m = 1, \dots, M$, be the m -th pre-specified kernel function and ϕ_m be the m -th associated feature mapping. Also, let \mathbf{K}_m^t be the $N_t \times N_t$ kernel matrix of the training data corresponding to the t -th task and calculated using the m -th kernel function, *i.e.* \mathbf{K}_m^t has elements $k_m(\mathbf{x}_i^t, \mathbf{x}_j^t)$, where $i, j = 1, \dots, N_t$. In this paper we consider the Multi-Task MKL framework that is formulated as follows:

$$\min_{\boldsymbol{\theta} \in \Psi(\boldsymbol{\theta})} \max_{\mathbf{a} \in \Omega(\mathbf{a})} \sum_{t=1}^T \bar{g}(\mathbf{a}^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t) \quad (6)$$

where $\mathbf{a} \triangleq [\alpha^1, \dots, \alpha^{T'}]' \in \mathbb{R}^N$, $N = \sum_{t=1}^T N_t$, and \mathbf{a}^t denotes the transpose of $\alpha^t \in \mathbb{R}^{N_t}$, $t = 1, \dots, T$. Similarly, $\boldsymbol{\theta}$ stands for the vector with elements θ_m^t .

The framework is able to incorporate various feature space sharing approaches by appropriately specifying the feasible region of the kernel combination coefficients $\boldsymbol{\theta}$. For example, for $T = 1$ it can specialize to L_p -norm MKL [12], where $p \geq 1$, by using $\Psi(\boldsymbol{\theta}) = \{\boldsymbol{\theta} : \|\boldsymbol{\theta}\|_p \leq 1, \boldsymbol{\theta} \succeq \mathbf{0}\}$. Obviously, the L_2 -norm MKL [11] is also covered by our framework. Additionally, it can express the Multi-Task MKL model in [18] that allows individual tasks to utilize their own task-specific space in conjunction with a shared space component. This

is achieved by specifying $\Psi(\theta) = \{\theta : \theta_m^t = \zeta_m + \gamma_m^t, \zeta_m \geq 0, \gamma_m^t \geq 0, \sum_{m=1}^M \theta_m^t = 1, \sum_{m=1}^M \sum_{t=1}^T \gamma_m^t \leq \beta\}$. Hence, our framework also subsumes the Common Space (CS) and Independent Space (IS) Multi-Task MKL models as special cases. The former is obtained by letting $\theta_m^t = \zeta_m, \forall t, m$. In this case, all tasks share a common kernel function determined by the coefficients ζ_m 's. Appropriate constraints can be added on ζ , such as the L_p -norm constraint. On the other hand, if we let $\theta^t \triangleq [\theta_1^t, \dots, \theta_M^t]'$ and add independent constraints on each θ^t , then we obtain the IS model, where each task occupies its own feature space.

To mention a final example, a Group-Lasso type regularizer is employed on the SVM primal-domain weights in [1], which leads to intricate optimization problems and algorithms. Our framework specializes to this problem by specifying $\Psi(\theta) = \{\theta : \theta \geq 0, (\sum_{t=1}^T \|\theta^t\|_p^q)^{1/q} \leq a, p \geq 1, q \geq 1\}$. By appropriately choosing the values of p and q , different level of group-wise and intra-group sparsity can be obtained. In this case too, using our framework leads to an easier formulation that can be solved in a much more straightforward fashion via our algorithm.

In the next section, we first transform Problem (6) to an equivalent SIP problem. Subsequently, we demonstrate the equivalency between general SIP problems and EPF-based problems. The latter result will allow us to cast Problem (6) as an EPF-based problem, which, as it turns out, can be easily solved.

3 Exact Penalty Function Method

In general, the min-max Problem (6) is not easy to solve. However, it can be transformed to an equivalent epigraph problem in the following SIP form:

$$\begin{aligned} & \min_{\omega \in \mathbb{R}, \theta \in \Psi(\theta)} \omega \\ & \text{s.t. } \sum_{t=1}^T \bar{g}(\alpha^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t) \leq \omega, \forall \mathbf{a} \in \Omega(\mathbf{a}). \end{aligned} \quad (7)$$

Before solving Problem (7), we first show an equivalence between EPF-based problems [21] and general SIP problems. This will eventually facilitate the development of an algorithm for solving Problem (7).

General SIP problem

Consider the general SIP problem

$$\begin{aligned} & \min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) \\ & \text{s.t. } g(\mathbf{a}, \mathbf{x}) \leq 0, \forall \mathbf{a} \in \Omega(\mathbf{a}); \\ & \quad l_u(\mathbf{x}) = 0, u = 1, \dots, U; \\ & \quad r_v(\mathbf{x}) \leq 0, v = 1, \dots, V. \end{aligned} \quad (8)$$

with U equality and V inequality constraints. Now, suppose f, g , the l_u 's and r_v 's are continuously differentiable. The feasible region of \mathbf{x} is determined by both the SIP constraint involving g and the regular equality (l_u 's) and inequality (r_v 's) constraints. It is not difficult to see that Problem (7) is a special case of Problem (8) by defining $\mathbf{x} \triangleq [\omega, \theta']'$, $\mathbf{a} \triangleq [\alpha^1, \dots, \alpha^T]'$, $f(\mathbf{x}) \triangleq \omega$, $g(\mathbf{a}, \mathbf{x}) \triangleq \sum_{t=1}^T \bar{g}(\alpha^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t) - \omega$, and letting the constraints l_u 's and r_v 's define $\Psi(\theta)$.

EPF-based problem

For fixed \mathbf{x} , assume that there are $N(\mathbf{x})$ local maxima $\mathbf{a}_i^* \in \Omega(\mathbf{a})$ of $g(\mathbf{a}, \mathbf{x})$ that satisfy $g(\mathbf{a}_i^*, \mathbf{x}) \geq -\eta$ for some $\eta > 0$ and $g(\mathbf{a}_i^*, \mathbf{x}) \neq g(\mathbf{a}_j^*, \mathbf{x}), \forall \mathbf{a}_i^*, \mathbf{a}_j^* \in \Omega(\mathbf{a})$. Denote this set of local maxima as $E(\mathbf{x}) \triangleq \{\mathbf{a}_i^*\}_{i=1}^{N(\mathbf{x})}$, and let $I(\mathbf{x}) \triangleq \{1, 2, \dots, N(\mathbf{x})\}$. Obviously, each \mathbf{a}_i^* depends implicitly on \mathbf{x} . Thus, if the Implicit Function Theorem conditions hold, there is a function \mathbf{a}_i of \mathbf{x} , such that $\mathbf{a}_i^* = \mathbf{a}_i(\mathbf{x})$ and we can define $h_i(\mathbf{x}) \triangleq g(\mathbf{a}_i(\mathbf{x}), \mathbf{x})$. Then, the EPF $P(\mathbf{x})$ introduced in [21] is defined in a neighborhood of \mathbf{x} as:

$$P(\mathbf{x}) \triangleq f(\mathbf{x}) + \nu \sum_{i \in I(\mathbf{x})} h_i(\mathbf{x})_+ \quad (9)$$

where $\nu > 0$ and $h_i(\mathbf{x})_+ \triangleq \max\{0, h_i(\mathbf{x})\}$. We refer the interested reader to [21] for more details about EPFs. Now, consider the EPF-based optimization problem:

$$\begin{aligned}
& \min_{\mathbf{x} \in \mathbb{R}^n} P(\mathbf{x}) \\
& \text{s.t. } l_u(\mathbf{x}) = 0, \forall u; \ r_v(\mathbf{x}) \leq 0, \forall v.
\end{aligned} \tag{10}$$

In the next theorem, we state that the general SIP Problem (8) can be solved by solving the EPF-based Problem (10).

Theorem 1. *Let f , g , l_u 's and r_v 's in Problem (8) and Problem (10) be continuously differentiable, and let the EPF function $P(\mathbf{x})$ be defined as in Equation (9). Suppose for fixed \mathbf{x} , there are finite number of $\mathbf{a}_s \in \Omega(\mathbf{a})$, $s = 1, \dots, S$, such that $g(\mathbf{a}_s, \mathbf{x}) = 0$. If $\hat{\mathbf{x}}$ is in the feasible region of Problem (8) and solves Problem (10), then $\hat{\mathbf{x}}$ is a Karush-Kuhn-Tucker (KKT) point of Problem (8).*

The proof of the above theorem is given in Section .1 of the Appendix. The EPF optimization provides a way to solve the proposed general SIP-type Problem (8); in addition, it also avoids involving the SIP constraint. Obviously, since our framework of Problem (7) is a general SIP problem, we can solve its equivalent EPF-based problem instead. In the next section we introduce a simple and easy to implement algorithm to solve the EPF optimization problem for our framework.

4 Algorithm

In this section, we focus on solving the EPF-based Problem (10) for our framework. Consider Problem (10), for which we define $\mathbf{x} \triangleq [\omega, \boldsymbol{\theta}']'$, $\mathbf{a} \triangleq [\boldsymbol{\alpha}^{1'}, \dots, \boldsymbol{\alpha}^{T'}]'$, $f(\mathbf{x}) \triangleq \omega$, $g(\mathbf{a}, \mathbf{x}) \triangleq \sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t) - \omega$, and let the constraints l_u 's and r_v 's define $\Psi(\boldsymbol{\theta})$. In order to solve Problem (10), a descent algorithm is suggested in [21]. During iteration k and given \mathbf{x}_k , $E(\mathbf{x}_k)$ is calculated, a descent direction \mathbf{d}_k is found, and the variables are updated as $\mathbf{x}_{k+1} = \mathbf{x}_k + \epsilon \mathbf{d}_k$. We remind the reader that the set $E(\mathbf{x})$ contains \mathbf{a}_i^* 's, which satisfy $g(\mathbf{a}_i^*, \mathbf{x}) \geq -\eta$. In [21] it is proven that, for sufficiently large ν and sufficiently small step length $\epsilon > 0$, the limit point of $\{\mathbf{x}_k\}$ is a KKT point of Problem (8), if the sequence $\{\mathbf{d}_k\}$ is bounded and the sequence $\{\mathbf{x}_k\}$ remains in a bounded region, in which f and g are doubly-differentiable functions with bounded second-order derivatives. In the next theorem, we show how to find the descent direction \mathbf{d}_k for our framework given \mathbf{x}_k at the k -th iteration.

Theorem 2. *Let $\bar{g}(\boldsymbol{\alpha}, \mathbf{K})$ be affine with respect to the individual entries of \mathbf{K} . Suppose it has a finite maximum and a finite number of local maxima with respect to $\boldsymbol{\alpha}$ in the feasible set $\Omega(\boldsymbol{\alpha})$. Let l_u 's and r_v 's be continuous and differentiable. Let $E(\mathbf{x})$ and $I(\mathbf{x})$ be as defined in Section 3, and $\mathbf{a}_{i,k} \triangleq [\boldsymbol{\alpha}_{i,k}^{1'}, \dots, \boldsymbol{\alpha}_{i,k}^{T'}]'$ $\in E(\mathbf{x}_k)$, $i \in I(\mathbf{x}_k) = \{1, \dots, N(\mathbf{x}_k)\}$. Consider the following $N(\mathbf{x}_k)$ problems, $\forall i \in I(\mathbf{x}_k)$:*

$$\begin{aligned}
& \min_{\boldsymbol{\theta}} \sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}_{i,k}^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t) \\
& \text{s.t. } l_u(\mathbf{x}) = 0, \forall u; \ r_v(\mathbf{x}) \leq 0, \forall v.
\end{aligned} \tag{11}$$

Let $\hat{\boldsymbol{\theta}}_i$, consisting of all elements $\hat{\theta}_{m,i}^t$'s, be the solution to the i -th problem. Also, let $i_0 = \arg \min_{i \in I(\mathbf{x}_k)} \max_{j \in I(\mathbf{x}_k)} \{\sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}_{j,k}^t, \sum_{m=1}^M \hat{\theta}_{m,i}^t \mathbf{K}_m^t)\}$ and $\hat{\omega} = \min_{i \in I(\mathbf{x}_k)} \max_{j \in I(\mathbf{x}_k)} \{\sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}_{j,k}^t, \sum_{m=1}^M \hat{\theta}_{m,i}^t \mathbf{K}_m^t)\}$. Finally, let $\hat{\mathbf{x}} = [\hat{\omega}, \hat{\boldsymbol{\theta}}_{i_0}']'$. Then, $\mathbf{d}_k = \hat{\mathbf{x}} - \mathbf{x}_k$ is a descent direction for the EPF-based Problem (10) when $\nu > 1$.

The proof of the above theorem is provided in Section .2 of the appendix. Note that, if \bar{g} is concave with respect to $\boldsymbol{\alpha}$, then $N(\mathbf{x}_k) = 1$. Thus, we have the following corollary.

Corollary 1. *Let $\bar{g}(\boldsymbol{\alpha}, \mathbf{K})$ be affine with respect to the individual entries of \mathbf{K} and concave with respect to $\boldsymbol{\alpha}$. Let the l_u 's and r_v 's be continuous and differentiable. Also, let $E(\mathbf{x})$ and $I(\mathbf{x})$ be defined as in Section 3 and $\mathbf{a}_k \triangleq [\boldsymbol{\alpha}_k^{1'}, \dots, \boldsymbol{\alpha}_k^{T'}]'$ $\in E(\mathbf{x}_k)$. Consider the problem*

$$\begin{aligned}
& \min_{\boldsymbol{\theta}} \sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}_k^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t) \\
& \text{s.t. } l_u(\mathbf{x}) = 0, \forall u; \ r_v(\mathbf{x}) \leq 0, \forall v.
\end{aligned} \tag{12}$$

with solution $\hat{\theta}$. Let $\hat{\omega} \triangleq \sum_{t=1}^T \bar{g}(\alpha_k^t, \sum_{m=1}^M \hat{\theta}_m^t \mathbf{K}_m^t)$ and $\hat{\mathbf{x}} \triangleq [\hat{\omega}, \hat{\theta}']'$. Then, $\mathbf{d}_k = \hat{\mathbf{x}} - \mathbf{x}_k$ is a descent direction for the EPF-based Problem (10) when $\nu > 1$.

Based on the above discussion, we provide Algorithm 1. Note that we focus on the case, where \bar{g} is concave, since it is the most common one in the context of kernel machines.

Algorithm 1 Algorithm for solving Problem (10)

Choose M kernel functions. Calculate the kernel matrices \mathbf{K}_m^t for the T tasks and the M kernels. Choose \bar{g} and $\Psi(\theta)$ based on the characteristics of the problem at hand. Randomly initialize $\theta_0 \succeq \mathbf{0}$. Initialize η and ϵ_0 to small positive values. $k \leftarrow 0$.

while not converged **do**

$\mathbf{a}_k \leftarrow \arg \max_{\mathbf{a} \in \Omega(\mathbf{a})} \sum_t \bar{g}(\alpha^t, \sum_m \theta_{m,k}^t \mathbf{K}_m^t)$

$\hat{\theta}_k \leftarrow \text{solve Problem (12) given } \mathbf{a}_k$

$\hat{\omega}_k \leftarrow \sum_t \bar{g}(\alpha_k^t, \sum_m \hat{\theta}_{m,k}^t \mathbf{K}_m^t)$

$\theta_{k+1} \leftarrow \theta_k + \epsilon_k(\hat{\theta}_k - \theta_k)$

$\omega_{k+1} \leftarrow \omega_k + \epsilon_k(\hat{\omega}_k - \omega_k)$

$k \leftarrow k + 1$

end while

There are several ways to choose the step length ϵ_k in each step k . As suggested in [21], one possibility is to choose ϵ_k as the largest element of the set $\{1, \beta, \beta^2, \dots\}$, for some β , $0 < \beta < 1$, such that

$$[P(\mathbf{x}_k + \epsilon_k \mathbf{d}_k) - P(\mathbf{x}_k)] / \epsilon_k G_k \geq \sigma \quad (13)$$

where P is the EPF given in Problem (10), σ is some constant that satisfies $0 < \sigma < 1$, and G_k is the directional derivative of P with respect to \mathbf{x} at the k -th step. It is not difficult to show that

$$G_k = f(\hat{\mathbf{x}}_k) + \nu g(\mathbf{a}_k, \hat{\mathbf{x}}_k)_+ - f(\mathbf{x}_k) - \nu g(\mathbf{a}_k, \mathbf{x}_k)_+ \quad (14)$$

where $\hat{\mathbf{x}}_k = [\hat{\omega}_k, \hat{\theta}_k']'$ and $\mathbf{x}_k = [\omega_k, \theta_k']'$. For details of G_k 's calculation, when \bar{g} is not concave with respect to α , we refer the reader to Section .2 of the Appendix.

4.1 Analysis

The advantages of our algorithm are multifold. First, the framework hinges on relatively few mild constraints that are typically met in practice. Specifically, we only assumed that \bar{g} is continuous and doubly differentiable with respect to α with bounded second-order derivatives and finite number of local maxima, whose value are also finite, and \bar{g} is affine with respect to the elements of \mathbf{K} . There is no need for \bar{g} to be concave with respect to α . Note also that all these constraints are met for the 4 examples given in Problems 2 through 5. Therefore, the framework and its associated algorithm may enjoy wide applicability.

Secondly, the maximization problem with respect to \mathbf{a} in the first step of the algorithm can be separated into T independent problems under the commonly-encountered setting, where the feasible regions of the α^t 's are mutually independent, such as in the formulations of [18] and [15]. Furthermore, in most situations, solving each of the T problems can be addressed via readily-available efficient optimizers; for example, in the case of SVM problems, one could use *LIBSVM* [2]. In some other cases, a closed-form solution could be used instead, whenever available, such as in the case of KRR.

Thirdly, the minimization problem (12) is easy to solve. Since \bar{g} is affine with respect to the individual entries of the kernel matrix, it is also affine in the entries of θ , which leads to the following optimization problem:

$$\begin{aligned} \min_{\theta} \mathbf{c}'\theta \\ \text{s.t. } l_u(\mathbf{x}) = 0, \forall u; r_v(\mathbf{x}) \leq 0, \forall v. \end{aligned} \quad (15)$$

where \mathbf{c} is a coefficient vector. For many practical models, the feasible regions of Problem (15) are such that a closed-form solution can be found. For example, in the case of an L_p -norm constrained (single-task) MKL

model, the feasible region is defined by the constraints $\|\boldsymbol{\theta}\|_p \leq 1, \boldsymbol{\theta} \succeq \mathbf{0}$, for which a closed-form solution can be found. Another example is the setting considered in [18], which is of the form of Problem (12) and becomes

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & \mathbf{c}'\boldsymbol{\theta} \\ \text{s.t.} \quad & \theta_m^t = \zeta_m + \gamma_m^t, \zeta_m \geq 0, \gamma_m^t \geq 0, \forall m, t; \\ & \sum_{m=1}^M \theta_m^t = 1, \forall t; \sum_{m=1}^M \sum_{t=1}^T \gamma_m^t \leq \beta. \end{aligned} \quad (16)$$

Regarding the just-stated problem, a block coordinate descent algorithm can be employed to optimize $\boldsymbol{\zeta} = [\zeta_1, \dots, \zeta_M]$ and $\boldsymbol{\gamma} = [\gamma^1, \dots, \gamma^T]'$. Each sub-problem features a linear objective function with an L_1 -norm constraint, and, hence, a closed-form solution can be found. Obviously, doing so is much simpler than relying on cutting-plane algorithms employed by the authors. Finally, a closed-form solution can be obtained for an $L_p - L_q$ -norm constraint as well. Proposition 1 offers closed-form solutions for problems featuring a linear objective with a L_p -norm or a $L_p - L_q$ -norm constraint.

Proposition 1. *Let $\mathbf{c} \triangleq [c_1, \dots, c_n]' \in \mathbb{R}^n$ be the concatenation of T vectors $\mathbf{c}^1 \in \mathbb{R}^{n_1}, \dots, \mathbf{c}^T \in \mathbb{R}^{n_T}$ with $\sum_{t=1}^T n_t = n$. Suppose that, for $t = 1, \dots, T$, each \mathbf{c}^t has at least one negative element. Similarly, let $\boldsymbol{\theta} \triangleq [\theta_1, \dots, \theta_n] \in \mathbb{R}^n$ be the concatenation of $\boldsymbol{\theta}^1 \in \mathbb{R}^{n_1}, \dots, \boldsymbol{\theta}^T \in \mathbb{R}^{n_T}$. The optimization problem*

$$\begin{aligned} \min_{\boldsymbol{\theta} \in \mathbb{R}^n} \quad & \mathbf{c}'\boldsymbol{\theta} \\ \text{s.t.} \quad & \boldsymbol{\theta} \succeq \mathbf{0}, \left(\sum_{t=1}^T \|\boldsymbol{\theta}^t\|_p^q \right)^{1/q} \leq a \end{aligned} \quad (17)$$

has closed-form solution

$$\begin{cases} \hat{\boldsymbol{\theta}}^t = \frac{\sigma^t (\tilde{\mathbf{c}}^t)^r}{\|\tilde{\mathbf{c}}^t\|_{r+1}^r}, \forall t, & p > 1, q > 1 \\ \hat{\boldsymbol{\theta}}^t = \frac{a (\tilde{\mathbf{c}}^t)^r}{\|\tilde{\mathbf{c}}^t\|_{r+1}^r} [t = t_0], \forall t, & p > 1, q = 1 \\ \hat{\boldsymbol{\theta}} = \frac{a (\mathbf{c}_*)^s}{\|\mathbf{c}_*\|_{s+1}^s}, & p = 1, q > 1 \\ \hat{\boldsymbol{\theta}} = a \mathbf{e}_j, & p = 1, q = 1 \end{cases} \quad (18)$$

where $\sigma^t \triangleq \frac{a \|\tilde{\mathbf{c}}^t\|_{r+1}^s}{(\sum_{t=1}^T \|\tilde{\mathbf{c}}^t\|_{r+1}^s)^{1/q}}$, $[t = t_0] = 1$, if $t = t_0$, and equals 0, if otherwise, $t_0 \triangleq \arg \min_t \{\|\tilde{\mathbf{c}}^t\|_{r+1}\}$, $r \triangleq \frac{1}{p-1}$, $s \triangleq \frac{1}{q-1}$, $\tilde{\mathbf{c}}^t$ is a vector with elements $[\max\{-c_m^t, 0\}]_{m=1}^M$ and $(\tilde{\mathbf{c}}^t)^r$ denotes element-wise exponentiation of vectors. Also, \mathbf{c}^* represents the concatenation of $\mathbf{c}_*^t \triangleq \mathbf{c}^t \circ \mathbf{e}_{i_t}^t, t = 1, \dots, T$, where \circ is element-wise multiplication of vectors and $\mathbf{e}_{i_t}^t \in \mathbb{R}^{n_t}$ is a vector, whose i_t -th entry is 1, while the remaining are 0, and $i_t \triangleq \arg \min_i \{c_i^t\}$. Moreover, \mathbf{e}_j is an all-0 vector except that its j -th entry equals 1, where $j \triangleq \arg \min_i \{c_i\}$. Finally, if $\exists t$ such that $\mathbf{c}^t \succeq \mathbf{0}$, then $\hat{\boldsymbol{\theta}}^t = \mathbf{0}$.

The proof of the above proposition is given in Section .3 of the Appendix. It is worth mentioning that, based on Proposition 1, true within-task sparsity is only achieved, when $p = 1$. Similarly, true sparsity across tasks is only obtained, when $q = 1$. Therefore, depending on the desired type of sparsity, different parameter settings should be applied. For example, if within-task sparsity is desired, an $L_1 - L_2$ norm with $p = 1$ and $q = 2$ can be utilized, as discussed in [6].

For a non-concave \bar{g} , but one that has a finite number of local maxima with respect to $\boldsymbol{\alpha}$ given \mathbf{x}_k , based on Theorem 2, we only need to solve $N(\mathbf{x}_k)$ problems that are similar to Problem (12), which in most cases have closed-form solutions, as stated earlier.

Based on the above analysis, it is not difficult to see that, when a closed-form solution can be found in the second step, the complexity of our algorithm in each iteration is dominated by the complexity of the solver of the first step, such as the SVM, SVDD solver, etc. The time complexity of *LIBSVM* for solving a SVM or SVDD problem is given in [2]. A KRR-based model, as stated earlier, can be solved in constant time. Therefore, if the second step has a closed-form solution, then the complexity of each iteration for

such model is $O(1)$. Unsurprisingly, we observed in practice that our algorithm is usually slower than the special algorithms that are tailored to each specific problem type. For example, for single-task MKL with an L_p -norm constraint as discussed in [24], a block coordinate descent algorithm is used, which is equivalent to our algorithm with step length equal to 1, when our algorithm is adapted to solve this model. Thus, since our algorithm initializes the step length to be a small value, it is not a surprise that our algorithm is slower. However, if the step length of our algorithm is initialized to 1, both of the two methods are exactly the same. Additionally, for solving the model that is proposed in [18], when 500 samples from the USPS data set are used for training, our algorithm takes on average 9 seconds to train the model, while the algorithm proposed in [18] takes 7 seconds. As expected, the advantages of our algorithm, namely its simplicity in terms of implementation and its generality are offset by a somewhat reduced computational efficiency, when compared to highly specialized algorithms that are designed to solve very specific problems.

5 The Partially Shared Common Space Model

To demonstrate the flexibility of our framework and the capability of the associated algorithm, in this section we propose a Partially-Shared Common Space (PSCS) model for Multi-Task MKL as a novel concrete instance of our framework. As stated earlier, in MTL, several tasks sharing a common feature representation are trained simultaneously. For Multi-Task MKL, it is a natural choice to let all tasks share a common kernel function by letting $\theta_m^t = \zeta_m, \forall t$ (see Section 2). However, in practice, there may be some problems, for which sharing a common space is not the optimal choice. For example, an MTL problem may include a few complex tasks, but also some much simpler ones. In this situation, it may be difficult to find a common feature mapping so that all tasks perform well. Therefore, it is meaningful to let complex tasks to use their own task-specific space, while allowing the remaining tasks to share a common space. Motivated by this consideration, we introduce the PSCS problem formulation shown below:

$$\begin{aligned}
& \min_{\boldsymbol{\theta}, \boldsymbol{\zeta}, \boldsymbol{\gamma}} \max_{\boldsymbol{\alpha} \in \Omega(\boldsymbol{\alpha})} \sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t) \\
& \text{s.t. } \theta_m^t = \zeta_m + \gamma_m^t, \forall m, t; \\
& \boldsymbol{\zeta} \succeq \mathbf{0}, \|\boldsymbol{\zeta}\|_p \leq 1, p \geq 1; \\
& \boldsymbol{\gamma} \succeq \mathbf{0}, (\sum_{t=1}^T \|\boldsymbol{\gamma}^t\|_p^q)^{1/q} \leq 1, p \geq 1, q \geq 1.
\end{aligned} \tag{19}$$

The feasible region of $\boldsymbol{\theta}$ is chosen to meet our objectives: the L_p -norm constraint controls the sparsity of the common-space coefficients $\boldsymbol{\zeta}$, while the $L_p - L_q$ -norm constraint controls the within-group and group-wise sparsity. Choosing $q = 1$ induces sparsity on $\boldsymbol{\gamma}$ (group-wise sparsity), which will force most tasks to share a common feature space.

Due to the complicated nature of the constraints, it is far from straightforward to devise a simple, tailored algorithm to solve the PSCS model. However, Algorithm 1 can be readily applied. In each iteration, T kernel machines are first trained. Next, the minimization with respect to $\boldsymbol{\theta}$ can be accomplished via block coordinate descent to optimize $\boldsymbol{\zeta}$ as a block and $\boldsymbol{\gamma}$ as another block. In addition, the closed-form solution for $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ can be easily determined by Proposition 1.

6 Experimental Results

In the following subsections, we experimentally evaluate our PSCS model on classification tasks. To apply our framework for classification tasks, the objective function \bar{g} is specified as the dual-domain objective function for SVM training:

$$\sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t) = \sum_{t=1}^T (\boldsymbol{\alpha}^{t'} \mathbf{1} - \boldsymbol{\alpha}^{t'} \mathbf{Y}^t (\sum_{m=1}^M \theta_m^t \mathbf{K}_m^t) \mathbf{Y}^t \boldsymbol{\alpha}^t) \tag{20}$$

Note that all kernel functions in our experiments are used in their normalized form as $\frac{k(\mathbf{x}, \mathbf{y})}{\sqrt{k(\mathbf{x}, \mathbf{x})k(\mathbf{y}, \mathbf{y})}}$.

6.1 A qualitative case study

To qualitatively illustrate the potential of our framework, in this subsection we apply our approach to the well-known *Iris Flower* classification problem, which we will recast as a MTL problem. The associated data set includes 150 patterns, each of which comes from one of three Iris flower classes: *Setosa*, *Versicolour* and *Virginica* (respectively, class 1, 2 and 3). Each of these 3 classes is represented by 50 samples and every sample has 4 attributes corresponding to the width and length of the flower’s sepal and pedal. We chose only two attributes, namely the sepal width and length, to form a 2-dimensional data set, such that the distribution of patterns and the resulting decision boundaries can be visualized. Note that each attribute is normalized to $[0, 1]$. We split the three-class problem into three binary classification tasks by employing a one-against-one strategy. Specifically, these are: task 1 (class 1 vs. class 2), task 2 (class 1 vs. class 3), task 3 (class 2 vs. class 3). The data sets of task 1 and 2 are linearly separable, so it is desirable to obtain classifiers which will produce linear or almost-linear decision boundaries. On the other hand, the data set of task 3 is not linearly separable. Intuitively, one would expect a reasonable solution to have tasks 1 and 2 share a common feature space, while allowing task 3 to be mapped into an alternative, task-specific feature space.

For the purpose of this experiment, we employed Linear kernels, Polynomial kernels of degree 2, and Gaussian kernels using a spread parameter value of 5. All 150 patterns were used for training. The p parameter was set to 2 for non-sparse kernel combination, because we want to see how different tasks affect the weights of each kernel function. Additionally, q is set to 1, because we want to achieve inter-task sparsity for γ , *i.e.* some tasks share a common space specified by ζ . The experimental results are shown in Table 1 and Figure 1.

Table 1: Learned coefficients for the *Iris* classification multi-task problem

| Kernel | ζ | γ^1 | γ^2 | γ^3 |
|------------|---------|------------|------------|------------|
| Linear | 0.1828 | 0 | 0 | 0.0295 |
| Polynomial | 0.9421 | 0 | 0 | 0.1976 |
| Gaussian | 0.2812 | 0 | 0 | 0.9333 |

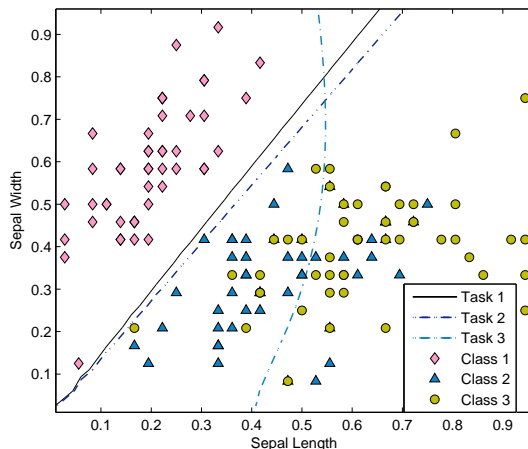


Figure 1: Decision boundaries of *Iris* classification multi-task problem.

It can be seen in Table 1 that all three elements in γ^1 and γ^2 are zero. This means that task 1 and 2 share a common space, which is specified by $\zeta = [0.1828, 0.9421, 0.2812]$. Obviously, in the common space, the combination of Linear and Polynomial kernels has a large weight, while the Gaussian kernel plays an insignificant role. The reason behind this is that the data for tasks 1 and 2 are both linearly separable,

so the SVM can easily find effective boundaries in the original space and the feature space implied by the Polynomial kernel. Also, we can observe that the polynomial kernel has larger weight than the linear kernel. This is because the data in the feature space induced by the polynomial kernel offers a larger margin than in the case of the original space. Note that, even though the polynomial kernel does not imply a linear feature map, the decision boundary is almost a straight line. This implies that the mapping corresponding to the polynomial kernel is almost linear in this particular case. Unlike tasks 1 and 2, we can see from the table that the task-specific feature space for task 3 has a large weight corresponding to the Gaussian kernel. This is because the data for task 3 are not linearly separable, and therefore it is difficult for the associated SVM to find a good decision boundary in the original space and the feature space implied by the Polynomial kernel. A large weight corresponding to the Gaussian kernel implies that the classifier was able to find a better decision boundary in its associated infinite-dimensional feature space. The relevant decision boundaries can be seen in Figure 1. As expected, the decision boundaries for tasks 1 and 2 are almost linear, while task 3 features a non-linear decision boundary.

6.2 Quantitative Analysis on Benchmark Problems

In this subsection we first evaluate our SVM-based PSCS method using 6 benchmark multi-class data sets obtained from the UCI repository [5]. Each associated recognition problem was cast as a multi-task classification problem by using the one-against-all approach. This is an effective method to test multi-task models; for example, refer to [9] and [26]. More specifically, we used the USPS Handwritten Digit (*USPS*), MNIST Handwritten Digit (*MNIST*), Wall-Following Robot Navigation (*Robot*), Statlog Shuttle (*Shuttle*), Statlog Vehicle Silhouettes (*Vehicle*), and Letter Recognition (*Letter*) data sets. For each data set, each class is represented by an equal number of samples. An exception is the original *Shuttle* data set that has seven classes, four of which are very poorly represented; for this data set we only chose data from the other three classes. Finally, for the *Letter* data set, we only chose the first 10 classes to avoid handling a large number tasks.

For all experiments, the kernel and algorithm parameter settings were held fixed. Twenty experiments were conducted for each setting and the average correct classification accuracy was recorded. Linear, Polynomial, and Gaussian kernels with spread parameter values $\{2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7\}$ were used. Regarding p , we held it fixed to 1.1, as we were not concerned with obtaining within-group sparsity. Cross-validation was employed to choose values for parameters C and q . We allowed C to vary over $\{1/27, 1/9, 1/3, 1, 3, 9, 27\}$ and q ranged from 1.0 to 2.0 with a step size of 0.1. We did not consider the case, where $q > 2$, since these values would yield a non-sparse group-wise γ vector, which was not deemed desirable.

The PSCS model was compared to the CS and IS methods, which have been described in Section 2. The experimental settings for the CS and IS methods were exactly the same as the settings used for PSCS, except that there was no parameter q to be tuned for them. We considered training set sizes of 2%, 5%, 20% and 50% of the original data set to study the effect of training set size on classification accuracy. The rest of the data were split in half to form validation and test sets.

We report the experimental results (average classification accuracy of 20 runs over randomly sampled training set) in Table 2, where the best performance is highlighted in boldface. To test the statistical significance of the differences between the best performing method and the rest, we employed a t-test to compare mean accuracies using a significance level of $\alpha = 0.05$. Furthermore, we use two superscripts for each PSCS-related result to convey information about statistically significant differences in performance. The first sign refers to the comparison of PSCS to CS and the second one to the comparison to IS. A ‘+’ sign means that the PSCS performance is significantly better, and a ‘=’ sign indicates that there is no statistically significant difference.

It can be seen from Table 2 that, for small training set sizes (2% and 5%), the PSCS method is almost always better in terms of classification accuracy compared to the CS and IS approaches and that most of the differences are statistically significant. In other words, there are learning tasks that, in practice, may benefit from partially sharing a common feature space. Additionally, for all three methods considered, as the training set size increases to 20% or higher, their performance tends to be similar for most data sets; PSCS seems to perform significantly better only for the *Vehicle* data set. It appears that, when the training set is large enough, each task independently can be trained well, while sharing a common space among tasks helps little to further enhance performance.

Table 2: Experimental comparison of PSCS to the CS, IS methods on six benchmark data sets in terms of percent correct classification accuracy.

| Robot | 2% | 5% | 20% | 50% |
|---------|----------------------------|----------------------------|----------------------------|----------------------------|
| PSCS | 75.58 ⁺⁺ | 82.21 ⁺⁺ | 90.02 ⁺⁺ | 92.27 ⁼⁼ |
| CS | 73.01 | 78.76 | 88.49 | 91.49 |
| IS | 73.09 | 78.91 | 87.86 | 91.76 |
| Vehicle | 2% | 5% | 20% | 50% |
| PSCS | 51.25 ⁼⁼ | 64.42 ⁺⁺ | 76.05 ⁺⁺ | 82.35 ⁺⁺ |
| CS | 48.78 | 61.24 | 75.58 | 81.47 |
| IS | 48.62 | 61.76 | 75.04 | 80.62 |
| USPS | 2% | 5% | 20% | 50% |
| PSCS | 52.78 ⁺⁺ | 68.41 ⁺⁺ | 87.03 ⁼⁼ | 93.34 ⁼⁼ |
| CS | 50.11 | 68.39 | 87.36 | 92.99 |
| IS | 46.52 | 63.95 | 86.91 | 92.78 |
| Shuttle | 2% | 5% | 20% | 50% |
| PSCS | 93.63 ⁺⁺ | 93.91 ⁺⁺ | 94.90 ⁺⁺ | 94.93 ⁼⁼ |
| CS | 91.80 | 91.84 | 94.20 | 94.90 |
| IS | 92.46 | 92.53 | 94.15 | 94.66 |
| Letter | 2% | 5% | 20% | 50% |
| PSCS | 50.19 ⁼⁼ | 67.43 ⁺⁺ | 82.65 ⁼⁼ | 89.23 ⁼⁼ |
| CS | 48.82 | 63.90 | 82.48 | 89.12 |
| IS | 51.35 | 64.21 | 82.99 | 89.16 |
| MNIST | 2% | 5% | 20% | 50% |
| PSCS | 55.15 ⁺⁺ | 70.59 ⁺⁺ | 84.38 ⁼⁼ | 89.92 ⁼⁼ |
| CS | 55.32 | 69.18 | 84.84 | 89.81 |
| IS | 51.29 | 66.59 | 84.34 | 90.13 |

In what follows we compare the PSCS method to the CS and IS approaches on two widely-used multi-task classification data sets, namely the *Letter* and *Landmine* data sets. The *Letter* data set¹ is a collection of handwritten words compiled by Rob Kassel of the MIT Spoken Language Systems Group. The associated MTL problem involves 8 tasks, each of which is a binary classification problem. The 8 tasks are: ‘C’ vs. ‘E’, ‘G’ vs. ‘Y’, ‘M’ vs. ‘N’, ‘A’ vs. ‘G’, ‘I’ vs. ‘J’, ‘A’ vs. ‘O’, ‘F’ vs. ‘T’ and ‘H’ vs. ‘N’. Each letter is represented by a 8×16 pixel image, hence by a 128 dimensional feature vector. The goal for this problem is to correctly recognize the letters in each task. For this data set, our intention was to compare the performance of the three models on large data sizes. Therefore, we chose 1000 samples from each class, so that each task had 2000 training samples. Note that, due to insufficient data for classes ‘J’, ‘H’ and ‘F’, our multi-task recognition problem considered only five of the eight tasks.

On the other hand, the *Landmine* data set² consists of 29 binary classification tasks. Each datum is a 9-dimensional feature vector extracted from radar images that capture a single region, which may contain a landmine field. The 9 features include four moment-based features, three correlation-based features, one energy ration feature and one spatial variance feature [25]. Tasks 1 – 15 correspond to regions with relatively high foliage, while the other 14 tasks correspond to regions that are bare earth or desert. The tasks entail different amounts of data, varying from 30 to 96 samples. The goal is to classify regions as ones containing landmine fields or not.

Table 3: Experimental comparison of PSCS to the CS, IS methods on multi-task data sets in terms of percent correct classification accuracy.

| Letter | 0.2% | 0.5% | 5% | 50% |
|----------|----------------------------|----------------------------|----------------------------|----------------------------|
| PSCS | 59.86 ⁼⁼ | 67.02 ⁺⁺ | 88.02 ⁼⁼ | 94.29 ⁼⁼ |
| CS | 56.52 | 65.39 | 87.68 | 94.31 |
| IS | 56.41 | 63.85 | 87.80 | 94.45 |
| Landmine | 20% | 30% | 40% | 50% |
| PSCS | 69.89 ⁺⁺ | 73.82 ⁺⁺ | 75.12 ⁼⁺ | 77.15 ⁼⁺ |
| CS | 67.24 | 71.62 | 75.04 | 76.96 |
| IS | 66.34 | 70.91 | 74.30 | 76.41 |

The experimental setting for these two problems were the same as previous experiments, except that we did not use 2% and 5% of the *Landmine* data set for training, due to the small size of the data set. Instead, we started from 20% and increased the training set size in steps of 10%. Also, for the *Letter* data set, in order to show results for a wider range of training set sizes, we chose 0.2%, 0.5%, 5% and 50% of the data set for training.

The experimental results are displayed in Table 3, from which it can be observed that the experimental results are consistent with the ones obtained in the previous subsection. The PSCS method can improve the classification accuracy significantly, when the training set size is relatively small, while in the case of large training sets, the improvements are not statistically significant. In the latter situation, all three methods perform similarly.

It is also worth mentioning that, by using our proposed algorithm, all three models have the same asymptotic computation complexity. Obviously, in each iteration, all three models solve the same T SVM problems followed by solving Problem (12). Obviously, the former step differs for each method; Problem (12) has a closed-form solution based on Proposition 1 for each of the models. Therefore, the computational complexity per iteration is common among them and is dominated by the complexity of solving the T SVM problems.

On balance, since PSCS has no worse asymptotic runtime complexity, when compared to the IS and CS approaches, and since it offers a performance advantage in cases of sample scarcity, the use of PSCS seems much preferable over the IS and CS formulations.

¹Available at: <http://multitask.cs.berkeley.edu/>

²Available at: <http://people.ee.duke.edu/~lcarin/LandmineData.zip>

7 Conclusions

In this paper, we proposed a Multi-Task Multi-Kernel Learning (Multi-Task MKL) framework, which is formulated as a min-max problem and which subsumes a broad class of kernel-based learning problems. We showed that our formulation can be optimized by solving an Exact Penalty Function (EPF) optimization problem. Subsequently, we derived a simple algorithm to solve it, which, for some frequently-used learning tasks, is able to leverage from existing efficient solvers or from closed-form solutions. The availability of this algorithm eliminates the need of using existing, potentially much sophisticated, algorithms or devising algorithms that are specifically tailored to the problem at hand.

In order to illustrate the utility of this novel framework and associated algorithm, we devised the Partially-Shared Common Space (PSCS) Multi-Task MKL model as a special case of our framework, which allows some tasks to share a common feature space, while other tasks are allowed to be appropriately associated to their own task-specific feature spaces. Results obtained from experimenting with a collection of classification tasks demonstrated performance advantages of the PSCS formulation, especially in the case, where the amount of training data is limited.

Acknowledgements

C. Li acknowledges partial support from National Science Foundation (NSF) grant No. 0806931 and No. 0963146. Moreover, M. Georgiopoulos acknowledges partial support from NSF grants No. 0525429, No. 0963146, No. 1200566 and No. 1161228. Finally, G. C. Anagnostopoulos acknowledges partial support from NSF grant No. 0647018. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. Finally, the authors would like to thank the anonymous reviewers of this manuscript for their time and helpful comments.

References

- [1] Jonathan Aflalo, Aharon Ben-Tal, Chiranjib Bhattacharyya, Jagarlapudi Saketha Nath, and Sankaran Raman. Variable sparsity kernel learning. *Journal of Machine Learning Research*, 12:565–592, February 2011.
- [2] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [3] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. l_2 regularization for learning kernels. In *UAI*, 2009.
- [4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.
- [5] A. Frank and A. Asuncion. UCI machine learning repository, 2010. Available from: <http://archive.ics.uci.edu/ml>.
- [6] J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. *ArXiv e-prints*, 2010. Available from: <http://arxiv.org/abs/1001.0736>, [arXiv:1001.0736](https://arxiv.org/abs/1001.0736).
- [7] Mehmet Gonen and Ethem Alpaydin. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268, 2011.
- [8] Laurent Jacob, Francis Bach, and Jean-Philippe Vert. Clustered multi-task learning: a convex formulation. In *NIPS*, 2008.
- [9] Tony Jebara. Multitask sparsity via maximum entropy discrimination. *Journal of Machine Learning Research*, 12:75–110, 2011.
- [10] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *ICML*, 2011.

- [11] Marius Kloft, Ulf Brefeld, Pavel Laskov, and Soren Sonnenburg. Non-sparse multiple kernel learning. In *NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, 2008.
- [12] Marius Kloft, Ulf Brefeld, Soren Sonnenburg, Pavel Laskov, Klaus-Robert Muller, and Alexander Zien. Efficient and accurate l_p -norm multiple kernel learning. In *NIPS*, 2009.
- [13] Gert R.G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.
- [14] Alain Rakotomamonjy, Francis R. Bach, Stephane Canu, and Yves Grandvalet. Simple-MKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.
- [15] Alain Rakotomamonjy, Remi Flamary, Gilles Gasso, and Stephane Canu. $l_p - l_q$ penalty for sparse linear and sparse multiple kernel multitask learning. *IEEE Transactions on Neural Networks*, 22:1307–1320, 2011.
- [16] Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. In *ICML*, 1998.
- [17] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, and Alex J. Smola. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.
- [18] Lei Tang, Jianhui Chen, and Jieping Ye. On multiple kernel learning with multiple labels. In *IJCAI*, 2009.
- [19] David M.J. Tax and Robert P.W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20:1191–1199, 1999.
- [20] Vladimir Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.
- [21] G. A. Watson. Globally convergent methods for semi-infinite programming. *BIT Numerical Mathematics*, 21:362–373, 1981.
- [22] Christian Widmer, Nora C. Toussaint, Yasemin Altun, and Gunnar Ratsch. Multi-task multiple kernel learning (MT-MKL). In *NIPS 2010 Workshop: New Directions in Multiple Kernel Learning*, 2010.
- [23] Xinxing Xu, I.W. Tsang, and Dong Xu. Soft margin multiple kernel learning. *IEEE Transactions on Neural Networks and Learning Systems*, 24:749–761, 2013.
- [24] Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, 2010.
- [25] Ya Xue, Xuejun Liao, Laurence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.
- [26] Leon Wenliang Zhong and James T. Kwok. Convex multitask learning with flexible task clusters. In *ICML*, 2012.

.1 Proof to Theorem 1

First, note that if $\hat{\mathbf{x}}$ solves Problem (8), then the $\hat{\mathbf{a}}_s$'s, which satisfy $g(\hat{\mathbf{a}}_s, \hat{\mathbf{x}}) = 0$ and $\hat{\mathbf{a}}_s \in \Omega(\mathbf{a})$, are local maxima of $g(\mathbf{a}, \hat{\mathbf{x}})$. By assumption, there is a finite number of such $\hat{\mathbf{a}}_s$'s, say, S , which means that the problem has only S active constraints. We can therefore provide the KKT necessary conditions for $\hat{\mathbf{x}}$ to be a solution of Problem (8):

KKT conditions for Problem (8) Let $\hat{\mathbf{x}}$ be a solution of Problem (8). Then, there exist $\hat{\lambda}_s$, $s = 0, \dots, S$; $\hat{\rho}_u$, $u = 1, \dots, U$; $\hat{\mu}_v$, $v = 1, \dots, V$, not all zero, such that

$$\hat{\lambda}_0 \phi_j(\hat{\mathbf{x}}) + \sum_{s=1}^S \hat{\lambda}_s \psi_j(\hat{\mathbf{a}}_s, \hat{\mathbf{x}}) + \sum_{u=1}^U \hat{\rho}_u \tau_{u,j}(\hat{\mathbf{x}}) + \sum_{v=1}^V \hat{\mu}_v \omega_{v,j}(\hat{\mathbf{x}}) = 0, \quad j = 1, \dots, n \quad (21)$$

where $\phi_j(\hat{\mathbf{x}})$, $\psi_j(\hat{\mathbf{a}}_s, \hat{\mathbf{x}})$, $\tau_{u,j}(\hat{\mathbf{x}})$, $\omega_{v,j}(\hat{\mathbf{x}})$ denote the quantities $\frac{\partial f(\mathbf{x})}{\partial x_j}|_{\mathbf{x}=\hat{\mathbf{x}}}$, $\frac{\partial g(\hat{\mathbf{a}}_s, \mathbf{x})}{\partial x_j}|_{\mathbf{x}=\hat{\mathbf{x}}}$, $\frac{\partial l_u(\mathbf{x})}{\partial x_j}|_{\mathbf{x}=\hat{\mathbf{x}}}$, $\frac{\partial r_v(\mathbf{x})}{\partial x_j}|_{\mathbf{x}=\hat{\mathbf{x}}}$ respectively.

This is a direct extension of the KKT conditions of the following standard SIP problem provided in [21]:

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) \text{ s.t. } g(\mathbf{a}, \mathbf{x}) \leq 0, \forall \mathbf{a} \in \Omega(\mathbf{a}). \quad (22)$$

Similarly, we can state the KKT conditions for the EPF-based Problem (10):

KKT conditions for Problem (10) Let $\hat{\mathbf{x}}$ be a solution of Problem (10). Then, there exist $\hat{\lambda}'_0$ and $\hat{\lambda}'_s$, $s \in I(\hat{\mathbf{x}})$; $\hat{\rho}'_u$, $u = 1, \dots, U$; $\hat{\mu}'_v$, $v = 1, \dots, V$, not all zero, such that

$$\hat{\lambda}'_0 \phi_j(\hat{\mathbf{x}}) + \sum_{s \in I(\hat{\mathbf{x}})} \hat{\lambda}'_s \psi_j(\hat{\mathbf{a}}_s, \hat{\mathbf{x}}) + \sum_{u=1}^U \hat{\rho}'_u \tau_{u,j}(\hat{\mathbf{x}}) + \sum_{v=1}^V \hat{\mu}'_v \omega_{v,j}(\hat{\mathbf{x}}) = 0, j = 1, \dots, n \quad (23)$$

In light of (21) and (23), we arrive at the conclusion.

.2 Proof to Theorem 2

Before delving into the details, we provide a sketch of the proof. The proof itself includes two major stages. In the first stage, we prove that $\mathbf{d}_k = \hat{\mathbf{x}} - \mathbf{x}_k$ is a descent direction, if $\hat{\mathbf{x}}_k$ is the solution to the following problem:

$$\begin{aligned} \min_{\mathbf{x}} f(\mathbf{x}) + \nu \sum_{i \in I(\mathbf{x}_k)} g(\mathbf{a}_{i,k}, \mathbf{x})_+ \\ \text{s.t. } l_u(\mathbf{x}) = 0, \forall u; r_v(\mathbf{x}) \leq 0, \forall v. \end{aligned} \quad (24)$$

Note that since \mathbf{x}_k is given in the k -th iteration and, therefore, is fixed, $I(\mathbf{x}_k)$ and the $\mathbf{a}_{i,k}$'s are also fixed. In the second stage, we prove that the $\hat{\mathbf{x}}$ described in the theorem is a solution to Problem (24), when $\nu > 1$. **Stage 1.** We need to prove that the directional derivative of the objective function of Problem (24) is negative, given the direction $\mathbf{d}_k = \hat{\mathbf{x}} - \mathbf{x}_k$. Since we only consider \bar{g} to be affine in the individual entries of the kernel matrix, $g(\mathbf{a}_{i,k}, \mathbf{x}_k)$ could be expressed as

$$g(\mathbf{a}_{i,k}, \mathbf{x}_k) = \sum_{t=1}^T [\sum_{m=1}^M \theta_{m,k}^t (\mathbf{b}_1^t(\mathbf{a}_{i,k})' \hat{\mathbf{K}}_m^t + b_2^t(\mathbf{a}_{i,k}))] - \omega_k \quad (25)$$

where $\mathbf{b}_1^t(\cdot)$ is a vector-valued function, $b_2^t(\cdot)$ is a scalar function and $\hat{\mathbf{K}}_m^t$ is a vector with elements $k_m(\mathbf{x}_i^t, \mathbf{x}_j^t)$, $i, j = 1, \dots, N_t$. If we define $I_+(\mathbf{x}) \triangleq \{i \in I(\mathbf{x}), g(\mathbf{a}_i, \mathbf{x}) > 0\}$, $I_0(\mathbf{x}) \triangleq \{i \in I(\mathbf{x}), g(\mathbf{a}_i, \mathbf{x}) = 0\}$, $I_-(\mathbf{x}) \triangleq \{i \in I(\mathbf{x}), g(\mathbf{a}_i, \mathbf{x}) < 0\}$, then the directional derivative can be calculated as

$$G(\mathbf{x}_k, \mathbf{d}_k, \nu) = \mathbf{d}_k' \phi(\mathbf{x}_k) + \nu \sum_{i \in I_0(\mathbf{x}_k)} [\mathbf{d}_k' \psi(\mathbf{a}_{i,k}, \mathbf{x}_k)]_+ + \nu \sum_{i \in I_+(\mathbf{x}_k)} \mathbf{d}_k' \psi(\mathbf{a}_{i,k}, \mathbf{x}_k) \quad (26)$$

Let $\mathbf{d}_k \triangleq \hat{\mathbf{x}} - \mathbf{x}_k$, $H_{i,m,k}^t \triangleq \mathbf{b}_1^t(\mathbf{a}_{i,k})' \hat{\mathbf{K}}_m^t + b_2^t(\mathbf{a}_{i,k})$ and $\mathbf{H}_{i,k}$ be the vector with elements $H_{i,m,k}^t$, $m = 1, \dots, M$, $t = 1, \dots, T$. Then, we obtain

$$\begin{aligned} G(\mathbf{x}_k, \mathbf{d}_k, \nu) &= (\hat{\omega} - \omega_k) \\ &+ \nu \sum_{i \in I_0(\mathbf{x}_k)} [\hat{\theta}' \mathbf{H}_{i,k} - \hat{\omega} - \theta_k' \mathbf{H}_{i,k} + \omega_k]_+ \\ &+ \nu \sum_{i \in I_+(\mathbf{x}_k)} (\hat{\theta}' \mathbf{H}_{i,k} - \hat{\omega} - \theta_k' \mathbf{H}_{i,k} + \omega_k) \end{aligned} \quad (27)$$

Note that $\theta_k' \mathbf{H}_{i,k} - \omega_k = g(\mathbf{a}_{i,k}, \mathbf{x}_k) = 0$, when $i \in I_0(\mathbf{x}_k)$. Then, we get

$$G(\mathbf{x}_k, \mathbf{d}_k, \nu) = f(\hat{\mathbf{x}}) + \nu \sum_{i \in I(\mathbf{x}_k)} g(\mathbf{a}_{i,k}, \hat{\mathbf{x}})_+ - f(\mathbf{x}_k) - \nu \sum_{i \in I(\mathbf{x}_k)} g(\mathbf{a}_{i,k}, \mathbf{x}_k)_+ \quad (28)$$

Since $\hat{\mathbf{x}}$ minimizes (24), we have that $G(\mathbf{x}_k, \mathbf{d}_k, \nu) \leq 0$.

Stage 2. Denote the objective function in (24) as

$$F(\omega, \boldsymbol{\theta}) = F(\mathbf{x}) = f(\mathbf{x}) + \nu \sum_{i \in I(\mathbf{x}_k)} g(\mathbf{a}_{i,k}, \mathbf{x})_+ \quad (29)$$

We first prove that in Problem (24), for $\nu > 1$ and fixed $\boldsymbol{\theta}$, the optimal solution to ω is $\hat{\omega} = \max_{i \in I(\mathbf{x}_k)} \sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}_{i,k}^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t)$, which yields $F(\hat{\omega}, \boldsymbol{\theta}) = \hat{\omega}$. To see that this is true, first consider another $\bar{\omega} > \hat{\omega}$, which implies that $F(\bar{\omega}, \boldsymbol{\theta}) = \bar{\omega} > \hat{\omega}$. On the other hand, if we consider $\bar{\omega} < \hat{\omega}$, then we have that

$$F(\hat{\omega}, \boldsymbol{\theta}) - F(\bar{\omega}, \boldsymbol{\theta}) = \hat{\omega} - \bar{\omega} - \nu \sum_{i \in I(\mathbf{x}_k)} \left(\sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}_{i,k}^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t) - \bar{\omega} \right)_+ \quad (30)$$

For simplicity, assume $\bar{\omega}$ is such that only one element in the summation over i not zero. Then,

$$F(\hat{\omega}, \boldsymbol{\theta}) - F(\bar{\omega}, \boldsymbol{\theta}) = \hat{\omega} - \bar{\omega} - \nu(\hat{\omega} - \bar{\omega}) \quad (31)$$

Since $\nu > 1$, the above quantity is negative, which implies that $F(\hat{\omega}, \boldsymbol{\theta}) < F(\bar{\omega}, \boldsymbol{\theta})$. When $\bar{\omega}$ is such that more elements in the summation over i are not zero, the rationale is similar. Therefore, we proved that for fixed $\boldsymbol{\theta}$, the previously defined $\hat{\omega}$ is the optimum.

Based on the just-stated fact, in order to minimize $F(\omega, \boldsymbol{\theta})$, we need to find a $\boldsymbol{\theta}$ such that $\max_{i \in I(\mathbf{x})} \sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}_i^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t)$ is minimized:

$$\min_{\boldsymbol{\theta}} \max_{i \in I(\mathbf{x}_k)} \sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}_{i,k}^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t) \quad (32)$$

This problem is equivalent to the following:

$$\begin{aligned} & \min_{\boldsymbol{\theta}} \lambda \\ & \text{s.t. } \sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}_{i,k}^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t) \leq \lambda, \forall i \in I(\mathbf{x}_k). \end{aligned} \quad (33)$$

Therefore, we only need to solve the $|I_k|$ problems $\hat{\boldsymbol{\theta}}_i = \arg \min_{\boldsymbol{\theta}} \sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}_{i,k}^t, \sum_{m=1}^M \theta_m^t \mathbf{K}_m^t), \forall i \in I(\mathbf{x}_k)$, and find a value, say $\hat{\boldsymbol{\theta}}_{i_0}$, that minimizes the quantity $\max_{j \in I(\mathbf{x}_k)} \{ \sum_{t=1}^T \bar{g}(\boldsymbol{\alpha}_{j,k}^t, \sum_{m=1}^M \hat{\theta}_{m,i_0}^t \mathbf{K}_m^t) \}$.

.3 Proof to Proposition 1

The results for $p = 1, q = 1$ and $\mathbf{c}^t \succeq \mathbf{0}$ are obvious. In what follows, we assume that there is at least one negative element in each \mathbf{c}^t . We first prove the result for $p > 1, q > 1$. In this situation, the problem can be written as

$$\begin{aligned} & \min_{\boldsymbol{\theta} \in \mathbb{R}^n, \boldsymbol{\sigma} \in \mathbb{R}^T} \mathbf{c}' \boldsymbol{\theta} \\ & \text{s.t. } \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}^t\|_p = \sigma^t, \forall t; \|\boldsymbol{\sigma}\|_q \leq a. \end{aligned} \quad (34)$$

Note that the above problem is equivalent to the following:

$$\begin{aligned} & \min_{\boldsymbol{\theta} \in \mathbb{R}^n, \boldsymbol{\sigma} \in \mathbb{R}^T} \mathbf{c}' \boldsymbol{\theta} \\ & \text{s.t. } \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}^t\|_p \leq \sigma^t, \forall t; \|\boldsymbol{\sigma}\|_q \leq a. \end{aligned} \quad (35)$$

since the optimum $\boldsymbol{\theta}^t$ for the previously-stated problem must occur on the boundary. The Lagrangian of Problem (35) takes the form of

$$L = \sum_{t=1}^T (\mathbf{c}' \boldsymbol{\theta}^t + \alpha^t (\|\boldsymbol{\theta}^t\|_p - \sigma^t) - \beta^t \boldsymbol{\theta}^t) + \gamma (\|\boldsymbol{\sigma}\|_q - 1) \quad (36)$$

where α and β are the Lagrangian multipliers corresponding to the inequality constraints. Setting its partial derivatives with respect to θ^t and σ to $\mathbf{0}$ and adding the complementary slackness conditions yields the following set of equalities:

$$\begin{cases} \mathbf{c}^t + \alpha^t \|\theta^t\|_p^{1-p} (\theta^t)^{p-1} - \beta^t = \mathbf{0}; \forall t \\ \gamma \|\sigma\|_q^{1-q} \sigma^{q-1} - \alpha = \mathbf{0}; \\ \alpha^t (\|\theta^t\|_p - \sigma^t) = 0; \forall t \\ \beta_m^t \theta_m^t = 0; \forall m, t \\ \gamma (\|\sigma\|_q - 1) = 0. \end{cases} \quad (37)$$

Note that α should not be $\mathbf{0}$, otherwise the minimum of L would be either minus infinity or 0, which are both trivial cases. Therefore, we have that

$$\begin{cases} \|\theta^t\|_p = \sigma^t; \forall t \\ \theta^t = \sigma^t [\frac{1}{\alpha^t} (\beta^t - \mathbf{c}^t)]^{\frac{1}{p-1}} \forall t \end{cases} \quad (38)$$

Considering (37) and the fact that $\alpha^t > 0$, as well as assuming that $\sigma^t > 0$, we have $\beta_m (\beta_m^t - c_m^t)^{\frac{1}{p-1}} = 0, \forall m$. Since $\beta_m^t \geq 0$, we conclude that

$$\begin{cases} \beta_m^t = 0 & \text{if } c_m^t \leq 0 \\ \beta_m^t = c_m^t & \text{otherwise.} \end{cases} \quad (39)$$

So, if we let $\tilde{\mathbf{c}}^t \triangleq [\tilde{c}_1^t, \dots, \tilde{c}_n^t]'$ and $\tilde{c}_m^t \triangleq \beta_m^t - c_m^t$, we obtain $\tilde{c}_m^t = \max\{-c_m^t, 0\}, \forall m$. We end up with $\theta^t = \sigma^t (\frac{\tilde{\mathbf{c}}^t}{\alpha^t})^{\frac{1}{p-1}} \propto (\tilde{\mathbf{c}}^t)^{\frac{1}{p-1}}$. Since θ^t needs to satisfy $\|\theta^t\|_p = \sigma^t$, we normalize $(\tilde{\mathbf{c}}^t)^{\frac{1}{p-1}}$ and get

$$\theta^t = \sigma^t \frac{(\tilde{\mathbf{c}}^t)^{\frac{1}{p-1}}}{\|\tilde{\mathbf{c}}^t\|_{\frac{p}{p-1}}} \quad (40)$$

In the previous derivation, we assumed that $\sigma^t \neq 0$. If $\sigma^t = 0$, we know immediately from (38) that $\theta^t = \mathbf{0}$. By the same rationale, we can show that $(\sigma^t)^{q-1} \propto \alpha^t = \|\tilde{\mathbf{c}}^t\|_{\frac{p}{p-1}}$. Noting that the optimal σ must occur on the boundary, we arrive at the solution

$$\sigma^t = \frac{a \|\mathbf{c}^t\|_{r+1}^s}{(\sum_{t=1}^T \|\mathbf{c}^t\|_{r+1}^{s+1})^{1/q}}. \quad (41)$$

When $q = 1$, (40) is still valid. Therefore, we substitute (40) into (35) to obtain:

$$\min_{\sigma} \sum_{t=1}^T \sigma^t \|\tilde{\mathbf{c}}^t\|_{r+1} \text{ s.t. } \sigma > \mathbf{0}, \sum_{t=1}^T \sigma^t \leq a \quad (42)$$

Note that the constraint $\sigma > \mathbf{0}$ is added, since $\sigma^t = \|\theta^t\|_p$. Obviously, the optimum σ is $\sigma^t = a[t = t_0]$, where $t_0 \triangleq \arg \min_t \{\|\tilde{\mathbf{c}}^t\|_{r+1}\}$.

When $p = 1$, we first optimize with respect to θ . By fixing σ , our problem can be split into T minimization problems:

$$\min_{\theta^t} (\mathbf{c}^t)' \theta^t \text{ s.t. } \theta^t \succeq \mathbf{0}, \sum_{i=1}^{N_t} \theta_i^t \leq \sigma^t \quad (43)$$

The solution to this problem is $\theta^{t*} = \sigma^t \mathbf{e}_{i_t}^t$. Substituting this result into Problem (35), optimizing with respect to σ and using an approach similar to the previous proof yields the desired result.